BAB II
TINJAUAN PUSTAKA

2.1 Optimasi

Optimasi adalah ilmu yang sudah berkembang sejak Newton di abad ke 17 menemukan cara menghitung akar. Dewasa ini banyak berkembang teknik-teknik baru untuk menyelesaikan masalah optimasi. Teknik-teknik tersebut antara lain *conic pogramming, semidefinite programming, semiinfinite programming* dan beberapa teknik metaheuristik. Permasalahan optimasi bisa dikategorikan berdasarkan variabel keputusan, fungsi objektif dan konstrain. Salah satunya yakni *multi-objective optimization*, dimana masalah optimasi mempunyai lebih dari satu fungsi objektif. (Budi dan Paul, 2011).

2.2 Perhitungan Aliran Daya

Optimasi aliran daya dilakukan dengan menentukan fungsi objektif terlebih dahulu. Fungsi objektif yang digunakan terdiri dari satu fungsi objektif *(single objective function)* atau lebih dari satu fungsi objektif (*multi-objective function*). Contoh fungsi objektif yang digunakan dalam optimasi aliran daya adalah rugi-rugi daya aktif, rugi-rugi daya reaktif, biaya pembangkitan, indeks keamanan tegangan, dan indeks keamanan saluran, serta profil tegangan. Metode perhitungan aliran daya yang digunakan dalam penelitian ini ialah metode *Newton-Raphson*. Pada penelitian R. F. S. Budi, et al. (2017), penyelesaian aliran daya menggunakan metode *Newton-Raphson* memberikan waktu komputasi yang lebih cepat jika dibandingkan dengan metode lainnya dengan tingkat ketelitian yang sama.

Langkah pertama dalam perhitungan aliran daya adalah dengan membentuk matriks admitansi bus (Ybus) dari parameter saluran distribusi. Jika jumlah bus pada sistem adalah N bus, maka matriks admitansi bus yang terbentuk berukuran NxN dengan elemen Yij adalah

$$Y_{ij} = |Y_{ij}| \angle\theta_{ij} = |Y_{ij}| \cos\theta_{ij} + j\,|Y_{ij}| \sin\theta_{ij} \qquad (2.1)$$

Tegangan pada bus ke-i diberikan dalam bentuk polar, yaitu:

$$V_i = |V_i| \angle \delta_i = |V_i| \left(\cos \delta_i + j \sin \delta_i\right) \quad (2.2)$$

Jika P_i dan Q_i adalah daya aktif dan daya reaktif yang terhubung ke sistem melalui bus *i*, maka:

$$P_i = \sum_{n=1}^{N} |Yin\ ViVn| \cos(\theta in + \delta n - \delta i) \quad (2.3)$$

$$Q_i = - \sum_{n=1}^{N} |Yin\ ViVn| \sin(\theta in + \delta n - \delta i) \quad (2.4)$$

Dengan menggunakan persamaan 2.3 dan 2.4 diperoleh persamaan 2.5 dan 2.6

$$P_i = |V_i|^2 G_{ii} + \sum_{\substack{n=1 \\ n \neq i}}^{N} |Yin\ ViVn| \cos\ (\theta in + \delta n - \delta i) \quad (2.5)$$

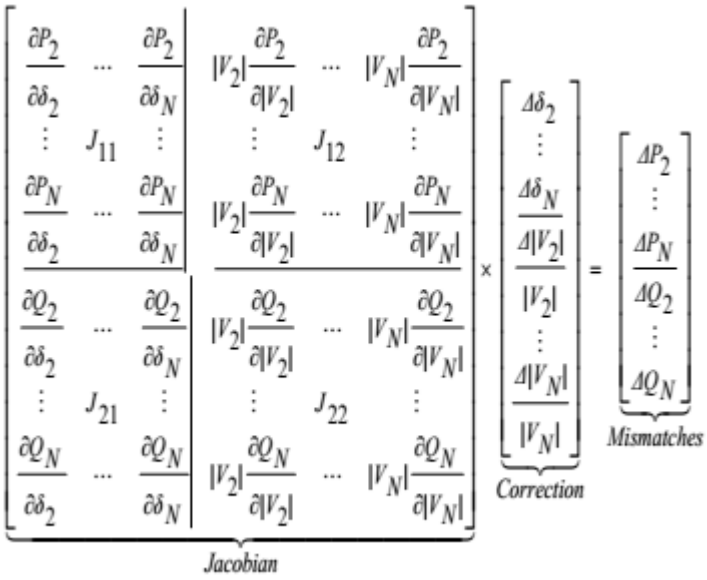
$$Q_i = - |V_i|^2 B_{ii} + \sum_{\substack{n=1 \\ n \neq i}}^{N} |Yin\ ViVn| \sin\ (\theta in + \delta n - \delta i) \quad (2.6)$$

Daya yang didapatkan dari sistem adalah daya yang direncanakan pada bus *i*($P_{i,sch}$, $Q_{i,sch}$). Sedangkan nilai daya yang diperoleh dari persamaan 2.5 dan 2.6 adalah daya hasil perhitungan ($P_{i,calc}$, $Q_{i,calc}$). Sehingga diperoleh *mismatch:*

$$\Delta P_i = P_{i,sch} - P_{i,calc} \quad (2.7)$$

$$\Delta Q_i = Q_{i,sch} - Q_{i,calc} \quad (2.8)$$

Jika $N = i$, maka dari persamaan 2.7 dan 2.8 akan diperoleh nilai tegangan V_i dan sudut fase tiap bus dengan menyelesiakan persamaaan mismatch dalam matriks dan vektor seperti yang ditunjukkan pada persamaan 2.9 (R. F. S. Budi et al., 2017).



(2.9)

Bus 1 merupakan *slack bus* yang tidak memiliki mismatch ΔPi dan ΔQi sehingga tidak dimasukkan dalam persamaan 2.9. Perhitungan aliran daya pada persamaan 2.9 dilakukan dengan cara menghitung Pi calc dan Qi calc dalam beberapa iterasi hingga didapatkan mismatch yang lebih kecil dari nilai toleransi. Selama iterasi akan didapatkan nilai tegangan dan sudut fase seperti pada persamaan 2.10 dan 2.11.

$$|V_i|^{(k+1)} = |V_i|^{(k)} + \Delta|V_i|^{(k)} = |V_i|^{(k)} \left(1 + \frac{\Delta|Vi|(k)}{|Vi|(k)}\right) \quad (2.10)$$

$$\delta_i^{(k+1)} = \delta_i^{(k)} + \Delta\delta_i^{(k)} \quad (2.11)$$

Iterasi akan berhenti ketika hasil dari persamaan 2.7 dan 2.8 lebih kecil dari toleransi yang ditetapkan. Hal ini menunjukkan bahwa hasil dari perhitungan aliran daya bukan merupakan proses optimasi (R. F. S. Budi, et al., 2016).

2.3 Optimasi Aliran Daya

Proses optimasi dilakukan untuk mendapatkan titik optimal berdasarkan fungsi objektif yang ditentukan dengan menambahkan metode optimasi dalam proses perhitungan aliran daya. Peneliti menggunakan metode optimasi *particle swarm optimization* dan *artificial bee colony* dalam penelitian ini.

Metode PSO banyak digunakan untuk mendapatkan solusi matematis yang rumit. Metode tersebut terinspirasi dari perilaku sosial binatang. Pergerakan dalam mencari solusi mengacu pada posisi partikel dalam sebuah populasi. Pergerakan tersebut dipengaruhi oleh lingkungannya yang berupa pembelajaran terhadap dirinya sendiri (*cognitive learning*) dan pembelajaran terhadap lingkungannya (*social learning*) (Risnandar, 2017). Oleh karena itu, perilaku kawanan burung akan didasarkan pada kombinasi dari 3 faktor sederhana, yakni kohesi (terbang bersama), separasi (jangan terlalu dekat), dan penyesuaian (*alignment*) (mengikuti arah bersama).

Pada algoritma ABC, koloni lebah terdiri atas tiga kelompok lebah, yakni lebah pekerja, lebah *onlooker*, dan lebah *scouts* (Prajapati, 2012).

2.4 Fungsi Objektif

Fungsi objektif yang digunakan dalam optimasi aliran daya pada penelitian ini adalah gabungan dari biaya pembangkitan dan rugi-rugi daya nyata pada saluran transmisi. Fungsi biaya pembangkitan bertujuan untuk meminimalkan biaya pembangkitan. Biaya pembangkitan dapat diperoleh dengan menggunakan persamaan 2.12 (Saukani, 2016).

$$F_i P_i = \sum_{i=1}^{n} \quad a_i + b_i P_i + c_i P_i^{\,2} \tag{2.12}$$

Fungsi rugi-rugi daya merupakan faktor utama dan mempengaruhi pengiriman daya yang optimal dari pembangkit. Persamaan rugi-rugi daya nyata generator yakni ditunjukkan pada persamaan 2.13.

$$P_L = G_k \left(V_i^{\,2} + V_j^{\,2} - 2V_iV_j \cos\left(\delta_i - \delta_j\right)\right) \tag{2.13}$$

Berdasarkan persamaan 2.13 dan 2.12 didapatkan fungsi objektif yang siap untuk dioptimasikan. Adapun persaamaan fungsi objektif tersebut adalah sebagai berikut:

$$F(x) = \sum_{i}^{n} \quad F_i P_i + 1000 * abs \left(\sum_{i}^{n} \quad P_i - P_D - \sum_{i}^{n} \quad P_L \right) \tag{2.14}$$

Nilai 1000 merupakan faktor skala dilantasi yang merupakan transformasi untuk dapat mengubah ukuran suatu titik. Sedangkan transformasi merupakan aturan secara geometris yang dapat menunjukkan cara suatu titik maupun bangun mampu berubah kedudukan dan ukurannya berdasarkan rumus tertentu.

P_i merupakan daya yang dibangkitkan, kemudian P_D merupakan daya beban pada pada pembangkit, dan P_L merupakan rugi-rugi daya.

Keluaran daya dan tegangan dari pembangkit memiliki batasan sebagai berikut:

$$V_{i(\min)} \leq V_i \leq V_{i(maks)} \quad (2.15)$$

$$P_{i(\min)} \leq P_i \leq P_{i(maks)} \quad (2.16)$$

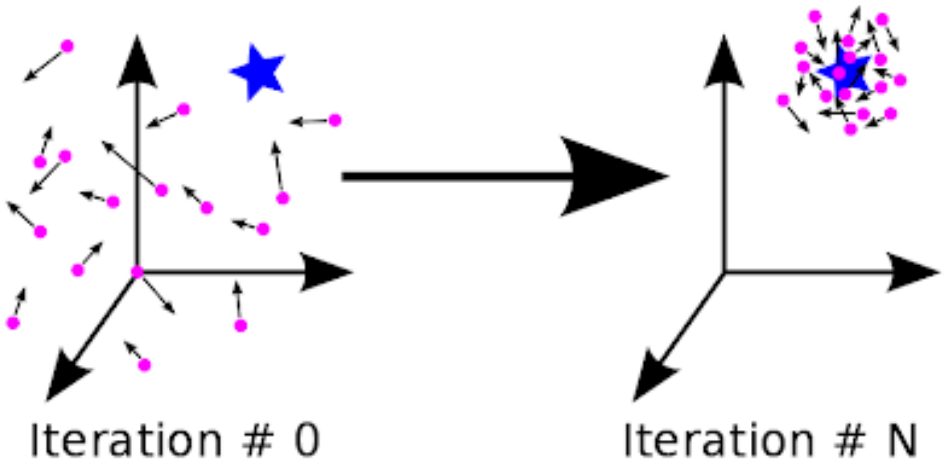
$P_{i(\min)}$ dan $P_{i(maks)}$ merupakan daya nyata minimum dan maksimum dari pembangkit ke-*i*.

2.5 Optimasi Menggunakan Particle Swarm Optimization

Particle swarm optimization adalah sebuah metode metaheuristik yang terinspirasi dari perilaku sosial binatang seperti kawanan burung dan ikan. Ketika mencari mangsa, kawanan burung tersebut bergerak bersama tanpa ada yang mengkoordinasi, namun pergerakannya memiliki derajat keteraturan tertentu.

Menurut Kennedy dan Eberhart (2001), metode ini menganut teori sosiokognitif yang dijelaskan dalam prinsip:

a. *Evaluate* : tiap individu memiliki kecenderungan untuk mengevaluasi rangsangan dan kemudian menyimpulkannya sebagai sesuatu yang menarik atau ditolak.
b. *Compare* : individu menilai dirinya sendiri dengan cara membandingkan dengan individu lain dan kemudian meniru individu yang dirasa lebih baik.
c. *Imitate* : mengimitasi merupakan cara efektif untuk belajar melakukan sesuatu. Meniru merupakan pusat sosialitas manusia dan berperan besar dalam penerimaan dan pembaharuan mental.



Gambar 2.1 Diagram Konsep *Particle Swarm Optimization* dengan *Single Global Minimum*

(*Sumber*: *Jonathan Becker*)

Secara matematis algoritma *particle swarm optimization* dapat dirumuskan sebagai berikut (Saukani, 2016) :

Rumus *update* kecepatan:

$$V_{ir+1} = w\,V_{ir} + c1.rand\,(P_{best_{ir}} - X_{ir}) + c2.rand(G_{best_{ir}} - X_{ir}) \quad (2.17)$$

Rumus *update* posisi :

$$X_{ir+1} = X_{ir} + V_{ir+1} \quad (2.18)$$

Rumus *weight* :

$$w_{it} = w_{max} - \frac{(w_{max} - w_{min})\,it}{it_{max}} \quad (2.19)$$

dengan:

X_{ir} : posisi kecepatan partikel saat ini

V_{ir} : kecepatan partikel saat ini

X_{ir+1} : posisi partikel iterasi selanjutnya

V_{ir+1} : posisi dan kecepatan partikel iterasi selanjutnya

c1 : konstanta kognitif

c2 : konstanta social acceleration

rand : nilai acak yang terdistribusi antara 0 dan 1

$P_{best_{ir}}$: posisi terbaik dari partikel itu sendiri

$G_{best_{ir}}$: posisi terbaik dari seluruh populasi yang ada

w_{max} : koefisien inersia *weight* maksimal

w_{min} : koefisien inersia *weight* minimal

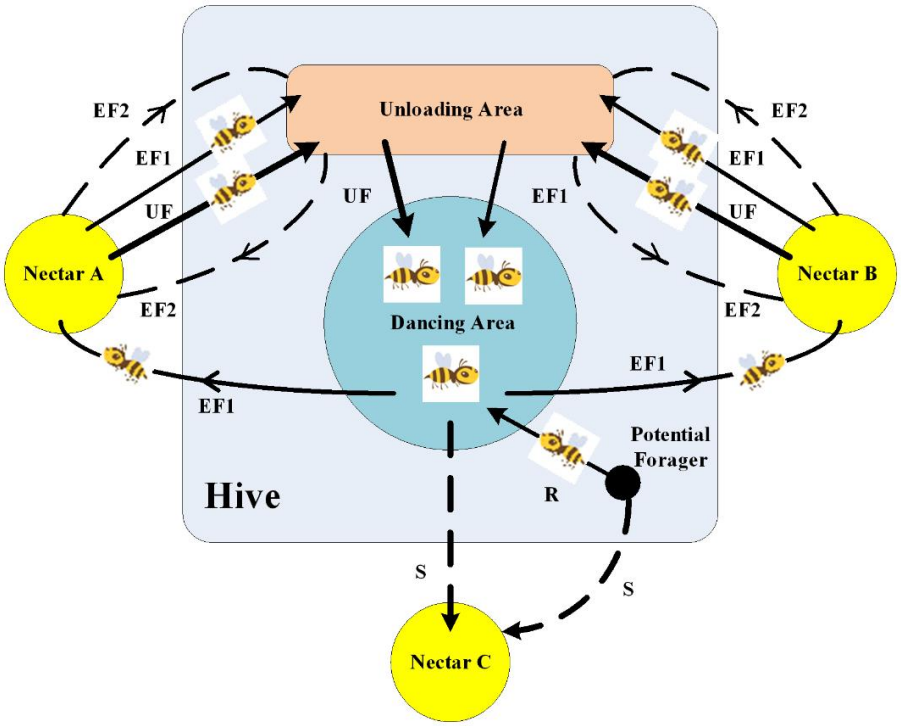
It : iterasi yang selalu berubah dari 1,2, … It_{max}

t_{max} : nilai maksimal dari iterasi yang digunakan

2.6 Optimasi Menggunakan Artificial Bee Colony

Algoritma *artificial bee colony* pertama kali diusulkan dengan mensimulasikan model simulasi *self-organization* dari lebah madu. Pada model ini, masing-masing lebah hanya melakukan satu tugas tunggal, namun melalui berbagai cara komunikasi informasi antara lebah. Seluruh koloni dapat menyelesaikan tugas seperti bangunan sarang, panen serbuk sari dan sebagainya (Duan, dkk., 2013).

Artificial bee colony bersama-sama mencari solusi optimal berdasarkan masalah yang diberikan. Tiap lebah buatan menghasilkan satu solusi untuk permasalahan ini. Terdapat dua fase dalam satu langkah algoritma *artificial bee colony* yaitu fase maju dan fase mundur (Danuri, 2013).



Gambar 2.2 Perilaku Lebah Madu Mencari Nektar

(*Sumber: Duan, dkk., 2013*)

Alur utama dari algoritma ABC yakni seperti di bawah ini (Hadi, 2015):

1. Inisialisasi sumber makanan awal secara acak

   $\theta_{ij} = \theta_{imin} + r(\theta_{imax} - \theta_{imin})$ (2.20)

   dengan:

   θ_i = posisi lebah pekerja

   i = 1 : SN (sumber makanan)

   j = 1 : N (jumlah koloni)

   r = nilai random [0,1]

2. Masing-masing lebah pekerja akan mencari sumber makanan baru yang dihasilkan dengan persamaan :

   $x_{ij}\,(t+1) = \theta_{ij}\,(t) + \phi\,(\theta_{ij}\,(t) - \theta_{kj}\,(t)$ (2.21)

   dengan:

   x = posisi lebah pekerja

t = jumlah iterasi

θ_k = lebah yang terpilih secara acak dan $k \neq i$

ϕ = urutan variabel acak [0,1]

3. Lebah *onlooker* memilih sumber makanan berdasarkan persamaan probabilitas sebagai berikut:

$$P_i = \frac{F(\theta i)}{\sum_{k=1}^{SN} F(\theta k)} \quad (2.22)$$

dengan:

P_i = probabilitas pemilihan

SN = jumlah sumber makanan

θ_i = posisi lebah pekerja

$F(\theta_i)$ = nilai *fitness*

4. Lebah pekerja yang meninggalkan sumber makanan berubah menjadi lebah *scout*. Lebah *scout* mencari sumber makanan baru yang dijabarkan dengan persamaan:

$$\theta_{ij} = \theta_{jmin} + r(\theta_{jmax} - \theta_{jmin}) \quad (2.23)$$